CSc 345 — Analysis of Discrete Structures
Fall 2024 (McCann)

http://cs.arizona.edu/classes/cs345/fall24/

Homework #4

(70 points)

Due Date: Thursday, December 5th, 2024, at the beginning of class

Write complete, legible answers to each of the following questions. (questions of the form "#.#" are exercises from the Shaffer PDF text). Show your work, when appropriate, for possible partial credit. **This is not a group project; do your own work!** *If you need help, remember that the TAs and I have office hours for just this eventuality, and you may also post general questions on Piazza.*

On the due date, by the start of class, submit a PDF containing your answers on gradescope.com. Be sure to assign pages to problems after you upload your PDF. (Need help? See "Submitting an Assignment" on https://help.gradescope.com/.) If you need to submit your solutions within the 24–hour late window, Gradescope will be configured to accept them. Solutions submitted after 3:30pm on the 6th will not be accepted. Want to be safe and submit your homework early? Please do! You can resubmit an updated PDF before the due date, if necessary.

1. [40 points] Binary Search, AVL and Splay Trees. (Shaffer Ch. 8 & 13.)

   For parts (a) and (b), we expect to receive typed (not hand–written!) versions of your methods. *The TAs will grade them by inspection, not by running them. We highly recommend that you actually code and test them, to be sure that they work. One way to do this is to add your code to the* `T07n02.java` *example program, available from the class web page.*

   (a) [10] 5.21 (in Java, of course).

   (b) [10] Write a boolean Java method `isAVL()` that, given a reference to the root node of a binary tree, returns true if the tree is a legal AVL tree, or false if it is not. You may call your answer to 1(a) (a.k.a. 5.21), and you may write additional helper methods if you wish.

   (c) [10] Build an AVL tree by inserting the following letters, in the order given, into the tree, and show the resulting tree after every third insertion (that is, show the tree after inserting Y, after inserting U, and after inserting N): $K, T, Y, V, W, U, G, E, N$

   (d) [5] Now assume that your final tree from part (c) is a splay tree. Search for U, and show the resulting tree after splaying.

   (e) [5] Using the tree from the end of part (d), search for E in that tree, and show the resulting tree after splaying.

2. [30 points] 2–3–... Trees. (Shaffer Ch. 10.)

   (a) [10] Construct a **2–3** tree from the sequence of data values $6, 9, 20, 8, 14, 27, 43, 22, 30$, and 19. Show the resulting tree, as well as your intermediate tree after the value 27 has been inserted.

   (b) [10] Construct a **2–3–4** tree from the sequence of data values used in part (a). When you need to promote, promote the second–largest value of the over–full node. Show the resulting tree, as well as your intermediate tree after the value 43 has been inserted.

   (c) [5] From the final 2–3–4 tree from part (b), delete 27 and show the resulting tree.

   (d) [5] From the tree at the end of part (c), delete 22 and show the resulting tree.